достижения асимптотически точного решения. Оценки погрешностей решений, полученных с помощью метода скользящего допуска, подтвердили обоснованность предложенного алгоритма.

Литература

1. Химмельблау Д. Прикладное нелинейное программирование [Текст] : пер. с англ. — М. : Мир, 1975. — 534 с.
2. Зеленцов Д. Г. Расчет конструкций с изменяющейся геометрией в агрессивных средах. Стержневые системы [Текст] / Д. Г. Зеленцов. — Днепропетровск : УГХТУ, 2002. — 168 с.
3. Короткая Л. И. Использование нейронных сетей при численном решении некоторых систем дифференциальных уравнений [Текст] / Л. И. Короткая // Восточно-Европейский журнал передовых технологий. — 2011. — № 3/4(51). — С. 24—27.

УДК 004.415.25

ВНУТРЕННЯЯ РЕАЛИЗАЦИЯ ОСНОВНЫХ МОДУЛЕЙ РАСПРЕДЕЛЕННЫХ СИСТЕМ НА ОСНОВЕ ТАБЛИЦ СОБЫТИЙ

О. Н. Кулешова
Аспирант*
Контактный тел.: 067-652-43-41
E-mail: v_olgo4ka@inbox.ru

Ю. К. Апраксин
Доктор технических наук, профессор*
Контактный тел.: (0692) 44-51-32, 050-661-55-64
E-mail: YKapraksin@ukr.net

Т. В. Волкова
Кандидат технически наук, доцент*
Контактный тел.: (0692) 45-62-51, 050-661-55-64
E-mail: volta2003@list.ru

*Кафедра кибернетики и вычислительной техники
Севастопольский национальный технический университет
ул. Университетская, 33, Севастополь, Украина, 99053

Проводиться аналіз та вибір моделі внутрішньої реалізації розподіленої системи на основі таблиць подій. Розробляється структура внутрішньої реалізації основних модулів системи: банка специфікацій таблиць подій та банку даних.

***Ключові** слова: таблиця подій, внутрішня реалізація, структура.*

Проводится анализ и выбор модели внутренней реализации распределенной системы на основе таблиц событий. Разрабатывается структура внутренней реализации основных модулей системы: банка спецификаций таблиц событий и банка данных.

***Ключевые слова:** таблица событий, внутренняя реализация, структура.*

Analysis and selection of the distributed system internal implementation model based on the event table are performed in the article. Structre of key system components (event table specification bank and databank) is developed in the article.

***Keywords:** event table, internal implementation, structre.*

Создание современных автоматизированных систем требует тщательной оценки альтернатив реализации с обязательным рассмотрением всех свойств, необходимых для эффективного функционирования этих систем. Табличные модели, значительно облегчая интерфейс проектировщик-система, дают возможность шире взглянуть на задачи проектирования систем управления и моделирования и на их решение. Таблицы событий [1] представляют собой универсальное средство для решения задач проектирования подобных систем.

Обеспечение возможностей проектирования и моделирования сложных технических систем с использованием таблиц событий предполагает разработку специальной распределенной программной системы. Кроме того, система на основе таблиц событий должна обеспечить возможность функционального расширения. Подобный подход предполагает включение в систему следующих подсистем: интерфейс пользователя, включающий подсистемы ввода и вывода; банк спецификаций, содержащий спецификации таблиц событий; банк данных; библиотеку процедур и функций; интерфейс проектировщика; подсистему проверки и коррекции таблиц событий; подсистему инициирования взаимодействия; подсистему внутренних переменных.

Для обеспечения функционирования системы в режиме проектирования, должны быть включены в систему, по крайней мере, следующие подсистемы: интерфейс проектировщика; банк спецификаций, содержащий спецификации таблиц событий; банк данных; библиотеку процедур и функций; подсистему проверки и коррекции таблиц событий.

Для обеспечения функционирования системы в режиме моделирования, должны быть включены в систему,

по крайней мере, следующие подсистемы: интерфейс пользователя, включающий подсистемы ввода и вывода; банк спецификаций, содержащий спецификации таблиц событий; банк данных; библиотеку процедур и функций; подсистему инициирования взаимодействия; подсистему внутренних переменных.

Структура системы представлена на рис. 1.

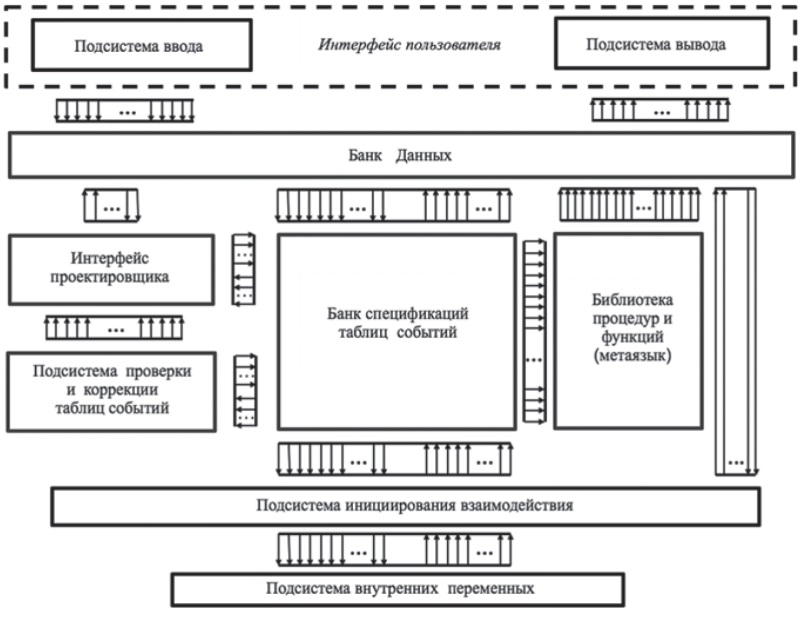

Рис. 1. Структура системы

Основными модулями системы являются банки спецификаций таблиц событий и данных.

Жизненный цикл распределенных систем включает в себя такие основные этапы как постановка задачи, спецификация, синтез, проектирование, тестирование, интеграция, квалификационные испытания, ввод в эксплуатацию и сопровождение [2].

Распределенная система должна обладать следующими важными характеристиками: возможностью совместного доступа к ресурсам, прозрачностью, открытостью, масштабируемостью, возможностью расширения, надежной защитой информации.

На этапе проектирования распределенной системы важна разработка схемы ее внутренней реализации. Для решения этой задачи широко применяется модель клиент-сервер [3], обеспечивающая высокую надежность, конфиденциальность данных, высокую эффективность обработки и хранения данных, высокое быстродействие, которое достигается за счет более качественного оборудования на сервере, возможность перераспределения задач среди вычислительных мощностей. Модель клиент-сервер содержит следующие уровни: уровень пользовательского интерфейса, уровень обработки и уровень данных.

В распределенной системе проектирования и моделирования на основе таблиц событий наиболее эффективной формой реализации уровня данных является реляционная база данных. Реляционная даталогическая модель данных [4] обеспечивает независимость данных. Данные организуются независимо от приложений так, чтобы изменения в организации данных не влияли на приложения, а приложения не оказывали влияние на организацию данных. Использование реляционной базы данных в модели клиент-сервер помогает отделить уровень обработки от уровня данных, рассматривая обработку и данные, независимо друг от друга.

Также, выбор реляционной модели, основывается на особенностях структуры информации содержащейся в таблицах событий и банке данных распределенной системы, таких как табличные представления, простота организации и необходимость поддержки целостности данных. Другие даталогические модели (объектно-ориентированная, многомерная, постреляционная) являются более громоздскими, сложными для понимания и не могут в полной мере обеспечить эффективность системы хранения информации.

Реляционные СУБД имеют ряд преимуществ, позволяющих реализовать основные требования распределенных систем:

- совместное использование данных;
- поддержка целостности данных;
- повышенная безопасность;
- повышение эффективности функционирования с ростом масштабов системы;
- возможность секционирования, база данных может быть физически распределена по разным компьютерам, но это будет скрыто от пользователя;
- наличие служб, вызов которых имеет стандартный синтаксис и семантику;
- надежное хранение и защита конфиденциальной информации, предотвращение несанкционированного доступа.

Несмотря на недостатки, такие как стоимость и дополнительные затраты на аппаратное обеспечение, использование реляционных СУБД в конкретных целях не вызывает сомнений.

Реализация в системе банка спецификаций таблиц событий, подразумевает, что все таблицы событий в системе имеют идентичную структуру, но содержат разные данные. Следовательно, таблицу событий в системе можно рассматривать как объект (или экземпляр), имеющий определенную структуру и хранимый определенным способом.

Таким образом, задача разработки способов хранения информации, содержащейся в таблицах событий, сводится к разработке типовой структуры, описывающей таблицу событий.

Предлагается следующая первичная спецификация таблицы событий (Event Table Specification — ETS):

ETS:: <ID,E, A, R, D, C, P, Cond, Act>,

где $E=\{E_i\}$; $i=\overline{1,m}$ — множество условий, описывающих параметры выбранной предметной области, $A=\{A_i\}$; $j=\overline{1,k}$ — множество действий, $SubT=\{SubT_i\}$; $j=\overline{1,k}$ — множество ссылок на таблицы решений, описывающих действия, $R=\{R_p\}$; $p=\overline{1,n}$ — множество решающих правил, называемых правилами решений, определяющими конкретные действия, при заданных условиях, $D=\{D_l\}$; $l=\overline{1,z}$ — множество решающих правил, определяющих действие «иначе», $C=\{C_q\}$, $q=\overline{1,n}$ — множество векторов условий, где $\{C_q\}=\{c_1^q, c_2^q, \ldots, c_m^q\}$,

$P=\{P_v\}$, $v=\overline{1,n}$ — множество действий, где $\{P_v\}=\{a_1^y, a_2^y, \ldots, a_k^y\}$, $Cond=\|c_{ix}\|, Act=\|a_{jx}\|$ — матрицы взаимосвязей множеств векторов данных и векторов действий. Столбцы c_x матрицы Cond задают возможные разбиения множества X на n непересекающихся классов, характеризующихся определенным действием Y, а столбцы a_x матрицы Act задают отношения между множествами условий E и множеством действий A.

$R_x=(c_x,a_x)$ называется решающим правилом, а элементы матрицы Cond и Act определяются следующим образом:

$$c_{ix}=\begin{cases}1, \text{ если условие } e_i - \text{истино},\\ 0, \text{если условие } e_i - \text{ложно},\\ \times, \text{ если условие } e_i - \text{безразлично};\end{cases}$$

$$a_{jx}=\begin{cases}a\in(1,\ldots,k), \text{ порядок выполнения действия},\\ \qquad\qquad \text{если правило } R_x \text{ приводит}\\ \qquad\qquad \text{к выбору решения } P_i,\\ 0, \text{ в противном случае}.\end{cases}$$

Таким образом, правило определяет действие, выполнение которого произойдет при конкретном сочетании результатов проверки условий.

Общий вид таблицы событий представлен в табл. 1.

Исходя из спецификации таблицы событий и правил разработки реляционных баз данных [5], структура таблицы событий представляет собой совокупность четырех таблиц базы данных, с учетом поддержки связей и целостности данных. Предлагаемая структура таблицы событий представлена на рис. 2.

Таблица id_TE_E хранит уникальный идентификатор условия (ID_E) и его описание на естественном языке (TEXT).

Таблица id_TE_A хранит уникальный идентификатор условия (ID_A), ссылку на таблицу событий, описывающую это действие, или конструкцию, использующую метаязык системы (SUBT), и описание действия на естественном языке (TEXT).

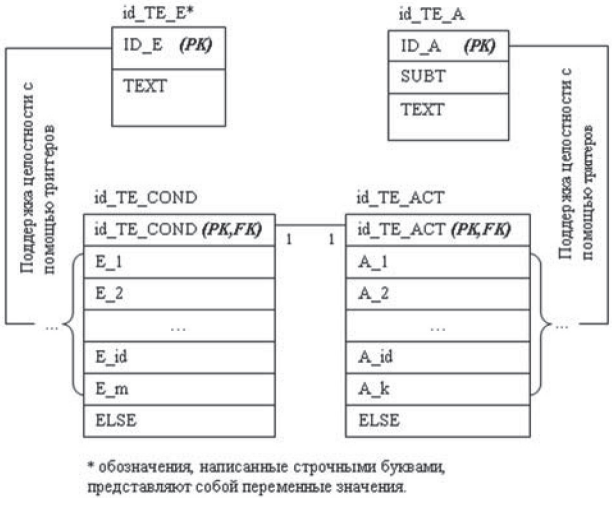

Рис. 2. Структура таблицы событий

Таблица id_TE_COND — динамическая таблица, в которой количество столбцов зависит от количества условий, заданных пользователем (проектировщиком). Таблица id_TE_COND описывает матрицу Cond. Столбец ID_COND представляет собой идентификатор вектора условий (C_x), столбцы E_id содержат в своих именах идентификаторы условий, столбец ELSE определяет, к какому типу относится вектор условий (относится к конкретно описанному сценарию (P_x), или описывает сценарий в случае «иначе» (P_{n+1})). Строка таблицы id_TE_COND представляет собой вектор условий (C_x).

Таблица id_TE_ACT — динамическая таблица, в которой количество столбцов зависит от количества действий, заданных пользователем (проектировщиком). Таблица id_TE_ACT описывает матрицу Act. Столбец ID_ACT представляет собой идентификатор сценария (P_x), столбцы A_id содержат в своих именах идентификаторы действий, столбец ELSE определяет, к какому типу относится сценарий (относится к определенному уникальным вектором условий (C_x), или описывает сценарий в случае «иначе» (D_l)).

Таблицы id_TE_COND и id_TE_ACT имеют связь 1:1. Это означает, что при добавление строки в таблицу id_TE_COND, строка с идентичным идентификатором будет добавлена в таблицу id_TE_ACT, и наоборот. В данном случае использование двух таблиц и связи 1:1, обусловлено необходимостью максимизировать допустимое количество условий и действий.

Связь и поддержка целостности данных между таблицами id_TE_E и id_TE_COND осуществляется с помощью триггеров. Использование триггера позволяет при добавлении строки в таблицу id_TE_E в таблицу id_TE_COND добавить столбец, содержащий в своем имени идентификатор условия id_E. При обновлении поля id_E в таблице id_TE_E, в таблице id_TE_COND будет обновлено имя столбца, содержащего в имени соответствующий идентификатор условия. При удалении строки из табли-

Таблица 1

Общий вид таблицы событий

ID			R_1	R_2	...	R_x	...	R_n	D					
			C_1	C_2	...	C_x	...	C_n	D_1	D_2	...	D_l	...	D_z
E	e_1		c_{11}	c_{12}	...	c_{1x}	...	c_{1n}	c_{1n+1}	c_{1n+2}	...	c_{1n+l}	...	c_{1n+z}
	e_2		c_{21}	c_{22}	...	c_{2x}	...	c_{2n}	c_{2n+1}	c_{2n+2}	...	c_{2n+l}	...	c_{2n+z}
	e_3		c_{31}	c_{32}	...	c_{3x}	...	c_{3n}	c_{3n+1}	c_{3n+2}	...	c_{3n+l}	...	c_{3n+z}
	...		...	...	...	...	...	...	...	...	...	...	...	...
	e_i		c_{i1}	c_{i2}		c_{ix}		c_{in}	c_{in+1}	c_{in+2}	...	c_{in+l}	...	c_{in+z}
	...		...	...	...	...	...	...	...	...	...	...	...	...
	e_m		c_{m1}	c_{m2}	...	c_{mx}	...	c_{mn}	c_{mn+1}	c_{mn+2}	...	c_{mn+l}	...	c_{mn+z}
			P_1	P_2	...	P_x	...	P_m	P_{n+1}					
A	A_1	$SubT_1$	a_{11}	a_{12}	...	a_{1x}	...	a_{1n}	a_{1n+1}					
	A_2	$SubT_2$	a_{21}	a_{22}	...	a_{2x}	...	a_{2n}	a_{2n+1}					
	A_3	$SubT_3$	a_{31}	a_{32}	...	a_{3x}	...	a_{3n}	a_{3n+1}					
	...	...	...	...	...	...	...	...	...					
	A_i	$SubT_i$	a_{i1}	a_{i2}	...	a_{ix}	...	a_{in}	a_{in+1}					
	...	...	...	...	...	...	...	...	...					
	A_k	$SubT_k$	a_{k1}	a_{k2}	...	a_{kx}	...	a_{kn}	a_{kn+1}					

цы id_TE_E из таблицы id_TE_COND будет удален столбец, содержащий в своем имени соответствующий идентификатор условия.

Связь и поддержка целостности данных между таблицами id_TE_A и id_TE_ACT осуществляется с помощью триггеров по аналогичной схеме.

Банк данных распределенной системы обеспечивает хранение основных данных, используемых для внутренней работы системы, а также хранение информации, поступающей на вход системы, и информации, которую система возвращает.

Предлагается следующая первичная спецификация банка данных распределенной системы на основе таблиц событий.

Данные в системе представлены в виде множества

$$\mathrm{Data} = \{\mathrm{Data}_\lambda\},\ \lambda = \overline{1,\mathrm{num}}.$$

Элемент данных представляется в виде

$$\mathrm{Data}_\lambda = \{\mathrm{UsedTE}^\lambda, \mathrm{FF}^\lambda, \mathrm{Type}^\lambda, \mathrm{Value}^\lambda\}.$$

$$\mathrm{UsedTE}^\lambda = \{\mathrm{UsedTE}^\lambda_\sigma\},\ \sigma = \overline{1,\mathrm{count}};\ \mathrm{count} \le \mathrm{quan}$$

— множество ссылок на таблицы событий, использующие этот элемент данных.

Ссылка на таблицу событий UsedTE_σ может быть представлена в виде кортежа

<UsedIDTE, UsedE, UsedA>

UsedIDTE — ID таблицы событий использующей элемент данных. $\mathrm{UsedE} = \{\mathrm{UsedE}_\xi\}$, $\xi = \overline{1,\mathrm{countE}}$ — множество ссылок на условия, использующих элемент данных. $\mathrm{UsedA} = \{\mathrm{UsedA}_\varsigma\}$, $\varsigma = \overline{1,\mathrm{countA}}$ — множество ссылок на действия, использующих элемент данных. Ссылка на действие UsedA_ς может быть представлена кортежем:

<UsedAID, RW>

UsedAID — ID действия, связанного с элементом данных. RW — флаг типа действия над элементом данных:

$$\mathrm{RW} = \begin{cases} 1, & \text{если} - \text{запись}; \\ 0, & \text{если} - \text{чтение}; \end{cases}$$

$\mathrm{FF}^\lambda = \{\mathrm{FF}^\lambda_\varepsilon\}$, $\varepsilon = \overline{1,\mathrm{funct}}$; $\mathrm{funct} \le \mu + \eta$ — множество ссылок на функции ввода и вывода, использующие этот элемент данных; Type^λ — тип элемента данных; Value^λ — значение элемента данных.

Исходя из спецификации банка данных и правил разработки реляционной базы данных [5], банк данных представляет собой совокупность трех таблиц базы данных, с учетом поддержки связей и целостности данных. Структура банка данных представлена на рис. 3.

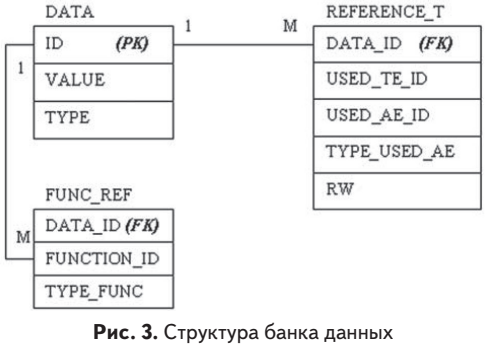

Рис. 3. Структура банка данных

Таблица DATA хранит уникальный идентификатор элемента данных (ID), тип элемента данных (TYPE), и непосредственное значение данных (VALUE).

Таблица REFERENCE_T хранит ссылки на таблицы событий использующие данные. Поле DATA_ID содержит идентификатор элемента данных. Поле USED_TE_ID предназначено для хранения ссылки на таблицу событий, использующую элемент данных. Поле USED_AE_ID предназначено для хранения ссылки на условие или действие в таблице событий. Поле TYPE_USED_AE для определения типа ссылки USED_AE_ID, может принимать два значения: 0 — если ссылка на условие, 1 — если на действие. Поле RW хранения информации о том, как условие или действие в таблице событий будет использовать элемент данных. Поле может принимать два значения: 0 — в случае чтения, 1 — в случае записи, при этом, если ссылка на условие, то значение поля RW может быть равно только 0, для действия возможны как чтение, так и запись.

Таблица FUNC_REF хранит ссылки на функции ввода-вывода, использующего элемент данных. Поле DATA_ID содержит идентификатор элемента данных. Поле FUNCTION_ID содержит ссылку на функцию ввода или вывода. Поле TYPE_FUNC содержит информацию, о том на какой тип функции ссылка — на функцию ввода или вывода; поле может принимать два значения: 0 — если ссылка на функцию ввода и 1 — если на функцию вывода.

Таблицы DATA и REFERENCE_T и таблицы DATA и FUNC_REF имеют связь 1 : M, чем обеспечивается связь и поддержка целостности данных.

Разработанная структура таблицы событий и банка данных, на основе реляционной базы данных дает возможность обеспечить независимость уровня данных от уровня обработки и уровня пользовательского интерфейса. Благодаря разработанной структуре данных, распределенная система проектирования и моделирования может быть легко модифицирована и расширена для обеспечения более широкого круга функциональных возможностей.

Литература

1. Кулешова О. Н. Спецификация распределенных систем на основе таблиц событий [Текст] / О. Н. Кулешова, Ю. К. Апраксин // Материалы 12-й Международной научно-технической конференции SAIT 2010. — К. : УНК «ИПСА» НТУУ «КПИ», 2010. — С. 450.
2. Таненбаум Э. Распределенные системы. Принципы и парадигмы [Текст] / Э. Таненбаум, М. ванн Стеен. — СПб. : Питер, 2003. — 877 с.
3. Кузнецов С. Д. Проектирование и разработка корпоративных информационных систем [Электронный ресурс]. Электронные текстовые данные (218 453 bytes) [Текст] / С. Д. Кузнецов. — М. : Центр Информационных технологий, 2001. — Режим доступа: http://www.citforum.ru/cfin/prcorpsys/index.shtml, свободный.
4. Хомоненко А. Д. Базы данных: Учебник для высших учебных заведений [Текст] / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев. — СПб. : КОРОНА принт, 2004. — 736 с.
5. Дейт К. Дж. Введение в системы баз данных 7-е издание [Текст] : пер. с англ. — М. : Издательский дом «Вильямс», 2001. — 1072 с.